R Package: GPTmodel

A. Polpo* and D. Sinha†

May 1, 2012

Abstract

The purposes of this text is to provide a "manual" for the GPTmodel package for R. We give some examples in how to use the main functions of the package.

Keywords: GPTmodel, R package, Generalized Power Transformation, reliability, error distribution, median estimation.

1 Introduction

We assume that the reader know about the reliability problem. We are treating the mainly with reliability, but almost all that we did here works fine for survival analysis. Probably, the switch the other "reliability" for "survival" will work for the ones that want use the package for survival analysis.

Also, we treat the problem of non-censor problems or a series system (competing risks). The case of parallel models are not developed yet (for this package). One remark is that the non-censor problem can be viewed as a particular case of series system, and a reliability/survival problem with censor can be modeled by a series system without problem.

Let C_1 and C_2 the random variables of the failure time of component 1 and 2. The failure time of the series sytem with two components is defined as $T = \min(C_1, C_2)$. The cause of failure it is known and defined by variable δ where $\delta = 1$ if was the first component that fail first and $\delta = 2$ if was the second.

polpo@ufscar.br, Department of Statistics – UFSCar and FSU

†*sinhad@stat.fsu.edu*, Department of Statistics – FSU

Translating the paragraph above to problem of right-censor data we just set the C_1 variable as the failure time without censor, C_2 as the censored time, and $\delta = 0$ if the time was censored and $\delta = 1$ if was not censored.

Example 1 Simulated data set of right-censor failure time. Let us say that the problem is about the experiment with 30 machines. We observe the failure time of this machines, but our experiment has a limit maximum of 6 week. Then all machines that not failed before 6 weeks will be considered as censored time. Also, we define the failure time by a random variable with Gamma distribution.

```
> data <- NULL
> time <- pmin(rgamma(30, 10, 2), rep(6, 30))
> delta <- rep(1, 30)
> for (i in 1:30) {
+     if (time[i] == 6)
+         delta[i] <- 0
+ }
> cbind(time, delta)
```

```
          time delta
 [1,] 3.071913     1
 [2,] 5.246828     1
 [3,] 5.319838     1
 [4,] 3.690964     1
 [5,] 5.561896     1
 [6,] 3.905557     1
 [7,] 6.000000     0
 [8,] 3.839161     1
 [9,] 4.043061     1
[10,] 2.435072     1
[11,] 4.849857     1
[12,] 6.000000     0
[13,] 4.581559     1
[14,] 4.778270     1
[15,] 5.899024     1
[16,] 3.805206     1
[17,] 4.958891     1
[18,] 4.023899     1
```

```
[19,] 4.757719     1
[20,] 4.692185     1
[21,] 2.780329     1
[22,] 4.767577     1
[23,] 4.765207     1
[24,] 5.852793     1
[25,] 6.000000     0
[26,] 6.000000     0
[27,] 6.000000     0
[28,] 2.748356     1
[29,] 6.000000     0
[30,] 3.245861     1
```

2 GPT Model

The GPT Model is defined as

$$g_\lambda(\log(T)) = g_\lambda(\theta(\mathbf{X})) + \varepsilon, \tag{1}$$

where the function $g_\lambda(\cdot)$ is the GPT defined by

$$g_\lambda(u) = \text{sign}(u)|u|^\lambda, \tag{2}$$

$\text{sign}(u) = 1$ if $u \geq 0$ and $\text{sign}(u) = -1$ if $u < 0$, $\lambda > 0$, $\theta(\mathbf{X})$ is a function of co-variables and the error has some distribution ($\varepsilon \sim F_\varepsilon$) with parameter ξ.

The first thing to do it is load the library.

```
> library("GPTmodel")
```

```
GPTmodel 0.1 beta loaded
```

The error distributions implemented in the GPTmodel package are Normal, t-Student, Cauchy, Double-Exponential and Logistic distributions. Also, we have the options to use the Reduced Weibull, Reflected Reduced Weibull, Double Weibull, Mixed Uniform and Mixed Central Uniform distributions. The variations of Weibull distributions (the three above) in ours studies does not give an improvement on reliability estimation, so its is needed more research to understand when these type of error distribution can be good. The cases of Mixed Uniform distributions have only educational purposes, the main question is about the size of parametric space.

Example 2 Density, reliability and hazard function for GPT model with different type of error distribution.

```
> x <- seq(0.1, 10, 0.1)
> plot(x, (1 - pgpt(x, l = 1, param = sqrt(2), beta = 1, dist = "norm")),
+     type = "l", lwd = 2, lty = 1, col = 2, ylim = c(0, 1), xlab = "t",
+     ylab = "R(t)", main = "Reliability", cex.lab = 1.2)

> plot(x, (dgpt(x, l = 1, param = sqrt(2), beta = 1, dist = "norm")),
+     type = "l", lwd = 2, lty = 1, col = 2, xlab = "t", ylab = "f(t)",
+     main = "Density", cex.lab = 1.2)

> plot(x, (hazard.gpt(x, l = 1, param = sqrt(2), beta = 1, dist = "norm")),
+     type = "l", lwd = 2, lty = 1, col = 2, xlab = "t", ylab = "h(t)",
+     main = "Hazard", cex.lab = 1.2)
```

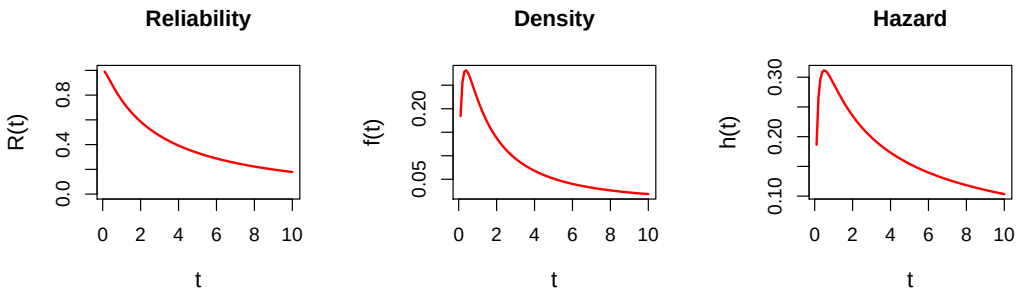

Figure 1: Density, reliability and Hazard functions
($\lambda = 1$; $\epsilon \sim \text{Normal}(0, 2^2)$; $\beta_0 = 1$).

```
> x <- seq(0.1, 10, 0.1)
> plot(x, (1 - pgpt(x, l = 1, param = 1, beta = 1, dist = "doubexp")),
+     type = "l", lwd = 2, lty = 1, col = 2, ylim = c(0, 1), xlab = "t",
+     ylab = "R(t)", main = "Reliability", cex.lab = 1.2)

> plot(x, (dgpt(x, l = 1, param = 1, beta = 1, dist = "doubexp")),
+     type = "l", lwd = 2, lty = 1, col = 2, xlab = "t", ylab = "f(t)",
+     main = "Density", cex.lab = 1.2)
```

```
> plot(x, (hazard.gpt(x, l = 1, param = 1, beta = 1, dist = "doubexp")),
+     type = "l", lwd = 2, lty = 1, col = 2, xlab = "t", ylab = "h(t)",
+     main = "Hazard", cex.lab = 1.2)
```

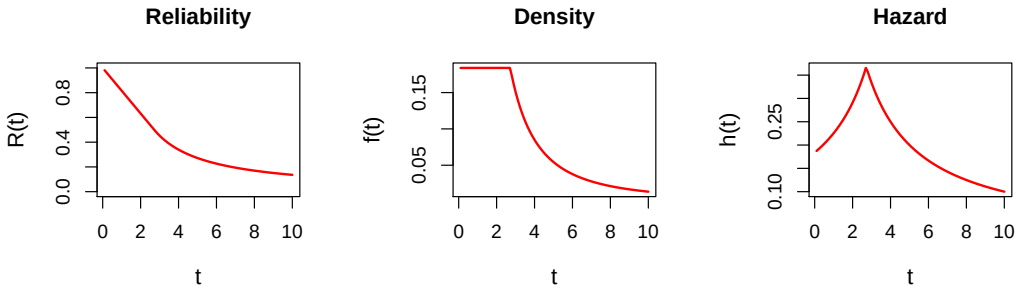

Figure 2: Density, reliability and Hazard functions
($\lambda = 1$; $\epsilon \sim$ doubexp(1); $\beta_0 = 1$).

2.1 Error distribution

Table **??** presents the available (and working) error distributions. See that for Normal distribution the parameter is σ (standard deviation) and not σ^2 (variance). Some functions use the following parameters: `"l"`: parameter of the GPT (λ), `"param"`: parameter of error distribution and `"beta"`: a vector with the βs parameters of $\theta(\mathbf{X})$. The main functions that uses these parameters are: `dgpt`, `pgpt`, `qgpt`, `rgpt` and `hazard.gpt`.

Table **??** give the "names" of the distribution to be used in the parameter `dist`.

We left these "not working" distribution options in the package because some one can be interested, principally in the density, cumulative distribution, quantile and generating functions (see functions `dRweib`, `dRRweib`, `doubweib`, `dmunif` and `dmcunif`) that all is working fine.

Table 1: Error distribution.

Distribution	Parameter	Density function ($f_\varepsilon(\epsilon\|\xi)$)
Normal	$\xi = \sigma$	$(2\pi\sigma^2)^{-1/2}\exp\{-\epsilon^2/(2\sigma^2)\}$
DoubExp	$\xi = b$	$(2b)^{-1}\exp\{-\|\epsilon\|/b\}$
t-Student	$\xi = \eta(d.f.)$	$\frac{\Gamma((\eta+1)/2)}{\Gamma(\eta/2)\sqrt{\pi\eta}}\left(1+\frac{\epsilon^2}{\eta}\right)^{-(\eta+1)/2}$
Cauchy	$\xi = c$	$[\pi c\,(1+(\epsilon/c)^2)]^{-1}$
Logistic	$\xi = s$	$\frac{\exp\{-\epsilon/s\}}{s\left[(1+\exp\{-\epsilon/s\})^2\right]}$

The parametric space for ξ is $(0, +\infty)$ in all cases.

Table 2: Error distribution.

Distribution	`dist`	work?
Normal	`"norm"`	Yes
DoubExp	`"doubexp"`	Yes
t-Student	`"t"`	Yes
Cauchy	`"cauchy"`	Yes
Logistic	`"logistic"`	Yes
Reduced Weibull	`"Rweib"`	Maybe
Refl. Red. Weibull	`"RRweib"`	Maybe
Double-Weibull	`"doubweib"`	Maybe
Mixed Unif	`"munif"`	almost sure that no!
Mixed Center Unif	`"mcunif"`	almost sure that no!

3 Estimation

The most important part is the parameter estimation of the GPT model. There are two procedures available: Maximum Likelihood Estimation (MLE) and Bayesian Estimation (BE). We did the estimation with both methods for the simulated data in Example **??**. Remember that we defined the observed failure time as `time` and the censor indicator δ as `delta`.

3.1 MLE

```
> gpt.mle <- estimation.mle(time = time, delta = delta, dist = "norm")
> gpt.mle
$kick
[1] 2.1 4.1 1.6

$par
[1] 1.115107 0.359380 1.543791

$std.error
[1] 1.10099214 0.49162334 0.06142312

$log.lik
[1] -46.43379

$error.dist
[1] "norm"

$AIC
[1] 98.86759

$AICc
[1] 99.79066

$BIC
[1] 103.0712

$KS
[1] 0.07174093

$method
[1] "Nelder-Mead"

$convergence
[1] TRUE

$run.time
```

```
  elapsed
0.1062167
```

Now we can perform the desired analysis. For example, we can compare the GPT model with the non-parametric Kaplan-Meier estimator.

```
> km <- survfit(formula = Surv(time, delta == 1) ~ 1)

> plot(km, ylab = "R(t)", xlab = "t: number of cycles (in thousands)",
+     main = "Reliability function (MLE)", conf.int = FALSE, lty = 1,
+     lwd = 1, xlim = c(min(time), max(time)))
> t <- seq(min(time), max(time), (max(time) - min(time) - 0.01)/1000)
> legend(2.6, 0.2, c("Kaplan-Meier", expression(textstyle(paste("GPT / ",
+     sep = "")) ~ epsilon ~ textstyle(paste("~", sep = "")) ~
+     Norm)), col = c(1, 2), lty = c(1, 1), cex = 1.1, lwd = c(1,
+     2), bg = "white")
> lines(t, 1 - pgpt(t, l = gpt.mle$par[1], param = gpt.mle$par[2],
+     beta = gpt.mle$par[3], dist = gpt.mle$error.dist), type = "l",
+     lwd = 2, col = 2, lty = 1)
```

The result is presented in the Figure **??**.

3.2 BE

It is important say that for Bayesian estimation we need to be concerned about the choose of the prior and the convergence of MCMC method. Because that, probably will be necessary some modifications in the code to adapt to your data. In the cases that we tested the simple change in `scale` parameter of the `estimation.bayes` was enough to use the Bayesian procedure. We are using a prior that reflect our knowledge about the parameter of the model. More details about that see Polpo and Sinha (XXXXXX).

```
> gpt.be <- estimation.bayes(kick = c(2, 4, 1.5), time = time,
+     delta = delta, dist = "norm", burn = 1000, jump = 10, size = 1000,
+     scale = 0.06)
```

Now we construct the graphic with the estimated reliability and the 95% credal interval of High Posterior Density of the reliability function.

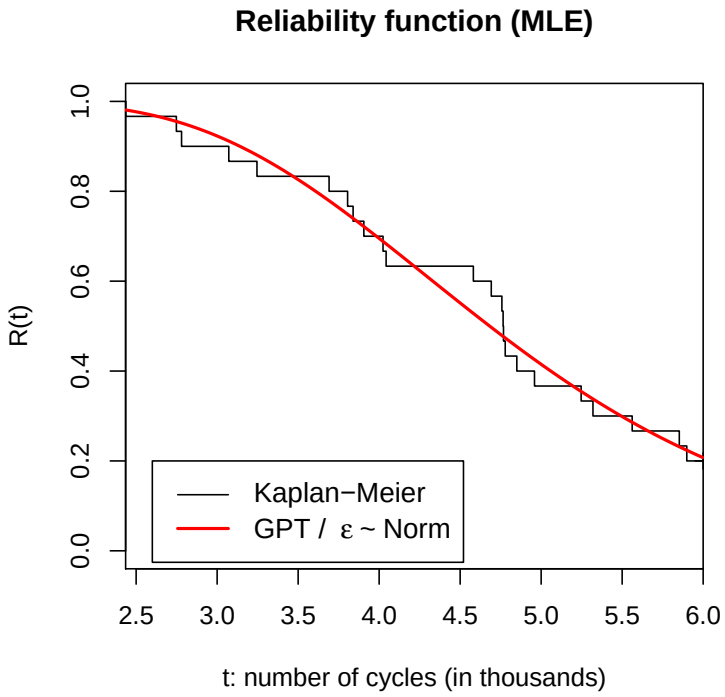

Figure 3: GPT model (MLE).

```
> plot(km, ylab = "R(t)", xlab = "t: number of cycles (in thousands)",
+     main = "Reliability function (BE)", conf.int = FALSE, lty = 1,
+     lwd = 1, xlim = c(min(time), max(time)))
> legend(2.6, 0.3, c("Kaplan-Meier", expression(textstyle(paste("GPT / ",
+     sep = "")) ~ epsilon ~ textstyle(paste("~", sep = "")) ~
+     Logistic), "95% HPD Interval"), col = c(1, 2, 2), lty = c(1,
+     1, 2), cex = 1.1, lwd = c(1, 2, 2), bg = "white")
> lines(gpt.be$time, gpt.be$survival[, 3], type = "l", lwd = 2,
+     col = 2, lty = 1)
> lines(gpt.be$time, gpt.be$survival[, 7], type = "l", lwd = 2,
+     col = 2, lty = 2)
> lines(gpt.be$time, gpt.be$survival[, 8], type = "l", lwd = 2,
+     col = 2, lty = 2)
```

The result is presented in the Figure **??**.

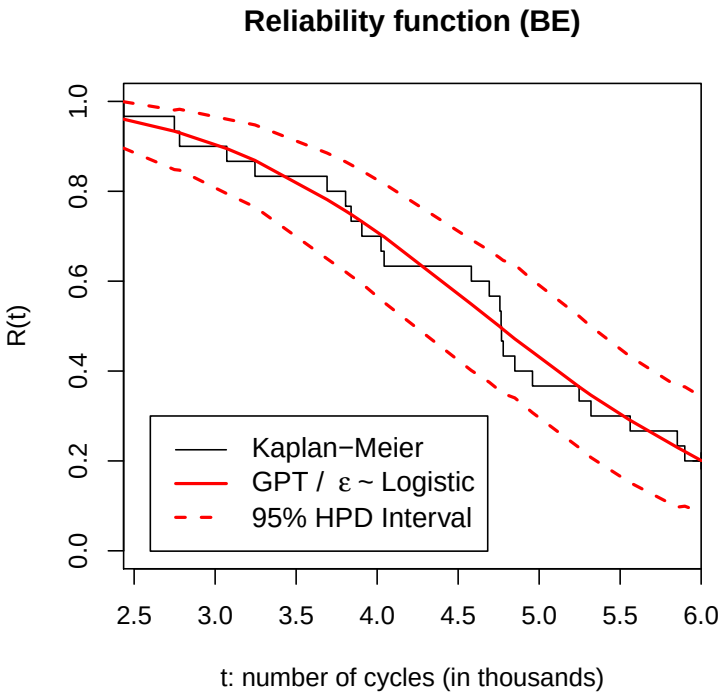

Figure 4: GPT model (BE).

4 Remarks

This "manual" content only the basics about the GPTmodel, we suggest to the reader take a look on the help pages of the GPTmodel package. More details about the GPTmodel can be obtained in Polpo and Sinha (XXXXXX).

A Article code

The source code used to "build" the article from Polpo and Sinha (XXXXX) is available with this package in the file `article.r`.